Sharing Cloud Computing Resources

Thomas Bonald, James Roberts

Sharing Cloud Computing Resources

Thomas Bonald[1] and James Roberts[2†]

[1] *Telecom ParisTech, Paris, France*
[2] *IRT System-X, Paris-Saclay, France*

We address the issue of sharing the CPU and RAM resources of a cloud computing system. Specifically, we argue that these resources should be shared according to proportional fairness instead of dominant resource fairness. The latter has recently been proposed as a desirable scheduling objective mainly because of its strategy-proofness property. We show that proportional fairness, which satisfies a weaker form of strategy-proofness that we refer to as scale-invariance, yields a much better resource utilization and outperforms dominant resource fairness in the realistic setting where jobs arrive and leave at random times.

Keywords: Cloud computing, resource sharing, dominant resource fairness, proportional fairness.

1 Introduction

Cloud computing systems serve extremely diverse jobs in terms of resource requirements (CPU, RAM, bandwidth, storage, etc.) and their performance critically depends on the way these resources are shared. Dominant resource fairness (DRF) has recently been proposed as a scheduling objective, with some desirable properties like Pareto-efficiency, sharing-incentive and strategy-proofness [GZH+11]. In this paper, we claim that resources should rather be shared according to proportional fairness (PF), long recognized as a desirable sharing objective for data flows in packet-switched networks [KMT98]. It turns out that PF has the same properties as DRF except for a weaker form of strategy-proofness that we refer to as scale-invariance. The key advantage of PF is better resource utilization that tends to decrease the mean completion time of jobs in the practically interesting case of a dynamic, randomly varying number of jobs.

2 Static sharing

We consider two types of resources, say CPU and RAM, with total capacities C and R, respectively. In this section, we assume that the number of jobs n is fixed and positive. There are K types of jobs. Each job of type k consists of a large number of independent tasks, each requiring c_k CPU units and r_k RAM units, with $c_k, r_k > 0$. All tasks cannot be served simultaneously in general. Denoting by n_k the number of jobs of type k and by φ_k the number of ongoing tasks of each job of type k, we have the capacity constraints:

$$\sum_{k=1}^{K} n_k \varphi_k c_k \leq C \quad \text{and} \quad \sum_{k=1}^{K} n_k \varphi_k r_k \leq R.$$

We assume tasks are infinitesimally small compared to the capacity of the cloud computing system. In this fluid model, we can take $C = R = 1$, normalize the resource requirements of each task accordingly and interpret the (now) real numbers $\varphi_1, \ldots, \varphi_K$ as volumes of tasks, with capacity constraints:

$$\sum_{k=1}^{K} n_k \varphi_k c_k \leq 1 \quad \text{and} \quad \sum_{k=1}^{K} n_k \varphi_k r_k \leq 1. \tag{1}$$

†The authors are members of the LINCS, Paris, France. See www.lincs.fr.

The resource allocation $\varphi = (\varphi_1, \ldots, \varphi_K)$ is said to be *Pareto-efficient* if at least one of these constraints, CPU or RAM, is attained. It is *sharing-incentive* if the resource shares are better than under a strictly fair allocation, *strategy-proof* if no job can increase its allocation by lying about its resource requirements and *scale-invariant* if the resource shares depend on the CPU and RAM requirements through their ratio only. Scale-invariance may be viewed as a weaker form of strategy-proofness in that no job can increase its allocation by scaling its resource requirements by merging tasks or dividing tasks into multiple subtasks.

Dominant resource fairness. The dominant resource of a job of type k is defined by $d_k = \max(c_k, r_k)$. DRF equalizes the dominant resource shares $\varphi_1 d_1, \ldots, \varphi_K d_K$. In view of the capacity constraints (1), this common resource share is given by:

$$\phi = \frac{1}{\max\left(\sum_{k=1}^{K} n_k \frac{c_k}{d_k}, \sum_{k=1}^{K} n_k \frac{r_k}{d_k}\right)}. \tag{2}$$

DRF is Pareto efficient, sharing-incentive and strategy proof [GZH+11]. Since ϕ depends on the CPU and RAM requirements through their ratio only, DRF is also scale invariant.

DRF does not utilize both CPU and RAM efficiently in general, due to its specific form imposed by the strategy-proofness property: only one of the two resources, that achieving the maximum in (2), is fully utilized. This is illustrated in Figure 1 where the CPU and RAM utilizations for $n = 10$ jobs of two types, with resource requirements $(c_1, r_1) = (1, 1/10)$ and $(c_2, r_2) = (1/10, 1)$, are shown with respect to the number of jobs of type 1, n_1. We observe that, except in the symmetric case $n_1 = n_2 = 5$, either the CPU or the RAM is under-utilized.

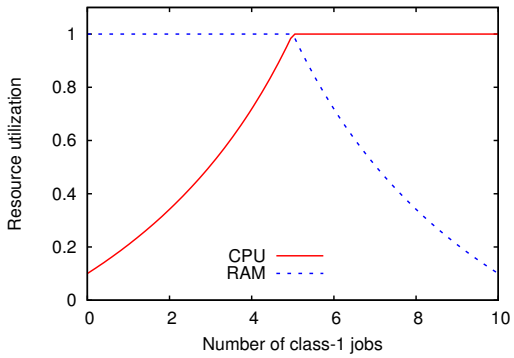


Fig. 1: Resource utilization under DRF ($n = 10$, $0 \le n_1 \le 10$).

Proportional fairness. PF is defined as the unique solution to the optimization problem:

$$\arg\max_{\varphi} \sum_{k=1}^{K} n_k \log \varphi_k, \tag{3}$$

under capacity constraints (1). By construction, it is Pareto efficient.

Proposition 1 *We have:*

$$\forall k = 1, \ldots, K, \quad (\alpha c_k + (1-\alpha) r_k)\varphi_k = \frac{1}{n}, \tag{4}$$

with $\alpha = 0$ *and* $\alpha = 1$ *if, respectively,*

$$\sum_{k=1}^{K} \frac{n_k}{n} \frac{c_k}{r_k} \le 1 \quad \textit{and} \quad \sum_{k=1}^{K} \frac{n_k}{n} \frac{r_k}{c_k} \le 1. \tag{5}$$

Otherwise, α is the unique solution over $(0,1)$ of equations:

$$\sum_{k=1}^{K} \frac{n_k}{n} \frac{r_k}{\alpha c_k + (1-\alpha) r_k} = 1 \quad \textit{and} \quad \sum_{k=1}^{K} \frac{n_k}{n} \frac{c_k}{\alpha c_k + (1-\alpha) r_k} = 1. \tag{6}$$

Proof. Let ν and ν' be the Lagrange multipliers associated with the CPU and RAM capacity constraints (1), respectively. By the Karush-Kuhn-Tucker theorem, PF is the unique allocation φ such that:

$$\forall k = 1, \ldots, K, \quad \frac{1}{\varphi_k} = \nu c_k + \nu' r_k,$$

with $\nu, \nu' \geq 0$,

$$\nu \left(\sum_{k=1}^{K} n_k \varphi_k c_k - 1 \right) = 0 \quad \text{and} \quad \nu' \left(\sum_{k=1}^{K} n_k \varphi_k r_k - 1 \right) = 0.$$

Summing these equalities yields $\nu + \nu' = n$. Letting $\alpha = \nu/(\nu + \nu')$ gives the desired result. □

In view of (4), $\max(\varphi_k c_k, \varphi_k r_k) \geq \frac{1}{n}$ for all k so that PF is sharing-incentive; since φ depends on the CPU and RAM requirements through their ratio only, PF is also scale invariant. On the other hand, PF is not strategy-proof [GZH+11].

The key advantage of PF over DRF is better resource utilization. In view of (5) and (6), PF fully utilizes *both* CPU and RAM whenever:

$$\sum_{k=1}^{K} \frac{n_k}{n} \frac{c_k}{r_k} > 1 \quad \text{and} \quad \sum_{k=1}^{K} \frac{n_k}{n} \frac{r_k}{c_k} > 1.$$

In the previous example of $n = 10$ jobs of two types, with resource requirements $(c_1, r_1) = (1, 1/10)$ and $(c_2, r_2) = (1/10, 1)$, this is always the case provided $n_1, n_2 > 0$.

3 Dynamic sharing

Although the previous example suggests that PF is more efficient than DRF, both allocations must be compared in the more realistic setting of a dynamic, randomly varying number of jobs to get insights into their actual performance. To this end, we now assume that jobs of type k arrive at rate λ_k and require the completion of σ_k tasks on average, each with mean service time τ_k. Then $\mu_k = 1/(\sigma_k \tau_k)$ would be the completion rate of a job of type k with exactly one ongoing task and $\varphi_k \mu_k$ is the completion rate of a job of type k in the considered fluid model. Under exponential statistical assumptions, the vector $\vec{n} = (n_1, \ldots, n_K)$ is a Markov process with birth rate λ_k and death rate $n_k \varphi_k \mu_k$ on component k. Insensitivity results suggest that the stationary distribution of this Markov process remains approximately the same under general statistical assumptions with the same means [BMPV06].

Since tasks of type-k jobs arrive at rate $\lambda_k \sigma_k$ and last τ_k seconds on average, the corresponding traffic intensity is $\rho_k = \lambda_k \sigma_k \tau_k = \lambda_k / \mu_k$. We deduce the respective CPU and RAM loads:

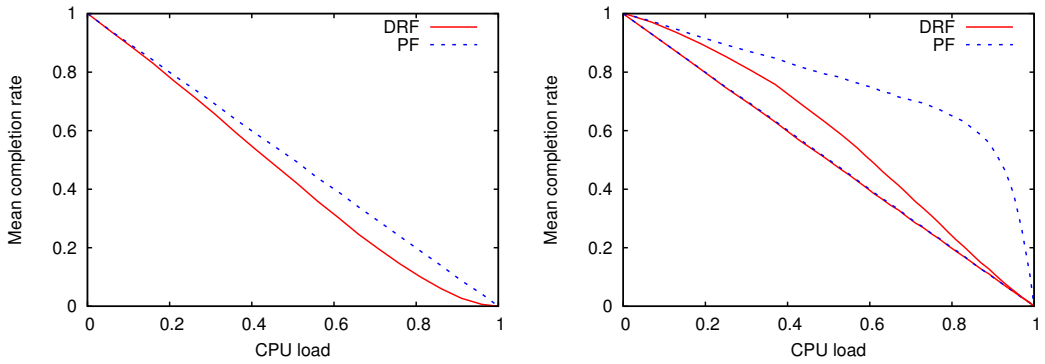
$$\sum_{k=1}^{K} \rho_k c_k \quad \text{and} \quad \sum_{k=1}^{K} \rho_k m_k.$$

Both loads must be less than 1 for the system to be stable. We are interested in the mean completion rate γ_k of jobs of type k, defined as the inverse of the mean completion time of jobs of type k. By Little's law, we have:

$$\gamma_k = \frac{\lambda_k}{\mathrm{E}(n_k)}.$$

Take the previous example of two types of job with resource requirements $(c_1, r_1) = (1, 1/10)$ and $(c_2, r_2) = (1/10, 1)$. We let $\mu_1 = \mu_2 = 1$ and denote by λ the total arrival rate. Figure 2 plots the mean

completion rates of jobs of each type, γ_1 and γ_2, against CPU load in the case of balanced traffic (a) $\lambda_1 = \lambda_2 = \lambda/2$ and in the case of unbalanced traffic (b) $\lambda_1 = 3\lambda/4$ and $\lambda_2 = \lambda/4$. The results are derived for each value of the CPU load from the simulation of 10^7 jumps of the Markov process $\vec{n}$. We observe that the performance of PF is slightly better than that of DRF in case (a) and significantly better for type 2 (top curves) in case (b), especially at high load. Note that in case (b), the CPU load is approximately three times higher than the RAM load; the number of type-1 jobs (which are mainly constrained by the CPU) tends to be much larger than the number of type-2 jobs (which are mainly constrained by the RAM) so that DRF under-utilizes the RAM (see Figure 1), to the detriment of type-2 jobs. PF, on the other hand, tends to utilize both CPU and RAM efficiently.



(a) Balanced traffic ($\lambda_1 = \lambda_2$) (b) Unbalanced traffic ($\lambda_1 = 3\lambda_2$)

Fig. 2: Mean completion rate against CPU load.

4 Conclusion

We argue that PF is preferable to DRF as a scheduling objective in cloud computing systems. PF satisfies the same properties as DRF except for a weaker form of strategy-proofness that we refer to as scale-invariance. More importantly, PF tends to utilize resources more efficiently, which results in lower job completion times in the realistic setting with random job arrivals and departures.

We plan in future work to confirm these results in more general and diverse traffic scenarios. We shall also consider various features of actual cloud computing systems, accounting notably for the fact that task resources must be allocated in a single machine, as studied in [PS13] for instance.

References

[BMPV06] T. Bonald, L. Massoulié, A. Proutière, and J. Virtamo. A queueing analysis of max-min fairness, proportional fairness and balanced fairness. *Queueing Syst. Theory Appl.*, 53(1-2):65–84, June 2006.

[GZH+11] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *Proceedings of USENIX*, 2011.

[KMT98] F. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49, 1998.

[PS13] Christos-Alexandros Psomas and Jarett Schwartz. Beyond beyond dominant resource fairness: Indivisible resource allocation in clusters. Tech Report Berkeley, 2013.